33rd Annual INCOSE international symposium
hybrid event
Honolulu, HI, USA
July 15 - 20, 2023

# A Case for Model-Based Systems Engineering in an Agile World and Principles for Growth

Daniel R. Call
Colorado State University
200 W. Lake Street
Fort Collins, CO 80523-6029
daniel.call@colostate.edu

Daniel R. Herber
Colorado State University
200 W. Lake Street
Fort Collins, CO 80523-6029
daniel.herber@colostate.edu



**Abstract.** Systems engineering enabled the development of some of the greatest engineering marvels of the 20th century. With the advent of the information age and software-centric systems, which drove a dramatic increase in system complexity, the practice of systems engineering has encountered growing pains. As organizations developing these software-centric systems work to adopt agile software development methods, the rigor of systems engineering work has been compromised. The linguistic relativity hypothesis suggests that the cognitive ability to engineer effective systems is developed and enhanced by the "language" of systems engineering. An improved understanding of what constitutes model-based systems engineering (MBSE) as opposed to systems modeling can improve MBSE outcomes. We put forth here that agile systems engineering, enabled by MBSE, is the bridge that will allow the principles and processes that have been developed through decades of research and experience to be applied to benefit modern systems.

## Introduction

While systems engineering (SE) has roots that reach back even further (Sillitto et al. 2018), it emerged as a discipline in the 1950s as a means to manage the complexity of large systems, like spacecraft and weapon systems, that began to be developed around that time (Hoban and Lawbaugh 1993). During the subsequent decades, the discipline matured as it became a field of research and study, standards emerged, and practitioners accumulated experience. The motivation of SE is to enable the realization, operation, maintenance, and disposal of systems (a collection of components and/or components and their relationships that provide a function that cannot be done by any of the constituent elements individually) that meets stakeholder needs and requirements by completing a series of defined technical and management processes throughout the system life cycle (Emes, A. Smith, and Cowper 2005). While it is not the purpose of this paper to recount the full history and purpose of SE as a discipline, it is important to recognize and appreciate its purpose and origins. While there is still room to improve to meet current and emerging challenges, SE is a mature

discipline that has evolved over decades of use and refinement and its principles are not obsolete. Rather, they are increasingly relevant to the successful delivery of modern systems.

Despite its mature foundation and past successes, SE has experienced significant growing pains in the information age. As a discipline that concerns itself with the specification and management of the components and relationships that constitute a system, its work has increased exponentially with the central role of software in modern systems. Figure 1 shows the exponential growth of software size in various domains. With this exponential growth, the legacy processes used to manage these systems through manually generated artifacts (documents, diagrams, spreadsheets, etc.) have quickly become untenable (Madni and Sievers 2018).

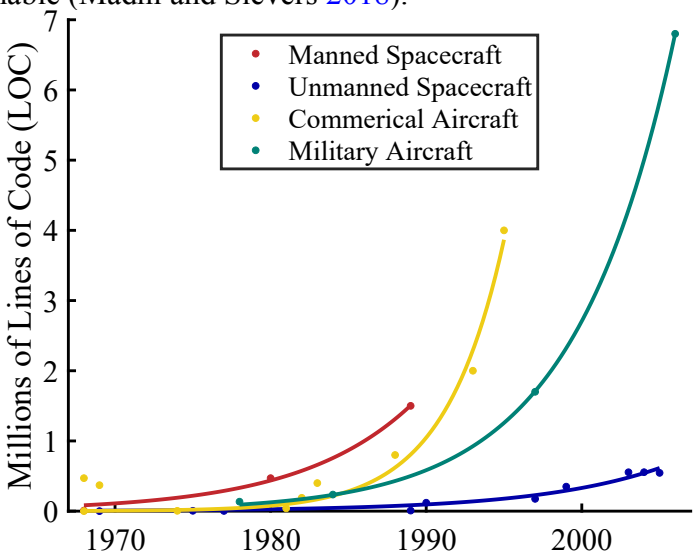


Figure 1. Historical trends in lines of code in various domains (data from Dvorak 2009; Potocki de Montalk 1993; Hagen and Sorenson 2013; Schenker, T. Smith, and Nichols 2022).

To illustrate the effects of this growth in system complexity on SE, consider a hypothetical, yet representative scenario of the software modernization efforts of a large weapon system. With the increasing complexity of subsystems and components, traditional, document-based SE approaches became too cumbersome to complete effectively and did not support required system development timelines within the organization (Kemp, Beasley, and Williams 2015). In addition to schedule and budget pressure, the challenges associated with this project's scope made it impossible to adequately manage SE efforts and artifacts which compromised SE rigor, resulting in a lower-quality product (Honour 2004; Rousseau 2018). The chief engineer and program manager recognized the need for a dramatic SE paradigm shift. Agile methods have revolutionized software development, and because of the software centric nature of the system of interest, it was proposed that agile methods be adopted at the system level and adapted for SE (Beck et al. 2001). Because agile methods were developed for software development based on the agile manifesto, there was no direct application of specific agile methods or processes to SE (Carson 2013). Instead, agile coaches taught the four key statements and accompanying 12 principles from the agile manifesto, as well as some techniques for scaling agile methods to a large enterprise (Laanti 2014).

These concepts were applied in the systems engineering context where practical. The most obvious

change was a move away from a waterfall development model where new product deliveries occurred every 3-5 years with many fully-formed capabilities and major development stages were completed for the entire system prior to moving on to the next step. Work was broken into increments with the understanding that capability would be delivered on a set cadence and partial capabilities that could deliver value to the user would be delivered before the full functionality of the capability was available. Cross-functional agile teams were formed, and development work commenced. Unfortunately, most of the changes were to the software development work within the organization, and there was no accompanying change in how the SE work was done to fit this agile approach. Systems engineers were forced to reduce the rigor of their processes to move faster and meet the new release cadence, to the point that they were often only reacting to work that was already in process. SE technical processes were pared down to the point that there was no one in the organization fulfilling the role of the traditional "Systems Engineer". In this particular organization SE died a slow, quiet death.

In an effort to address a legitimate problem, the inability of traditional SE methods to scale to the scope of modern systems, we all too often "throw the baby out with the bathwater" or abandon the valuable aspects of systems engineering in an effort to remove its undesirable qualities (Ammer 1997). This leads to inferior outcomes as SE is not only still relevant to modern systems but more important than ever. In this paper, we will highlight some aspects of SE that are particularly important to improving the practice of SE today.

The remainder of the paper is organized as follows. Section 2 addresses the importance of the continued use of the "language" of SE as cognitive abilities to complete SE tasks can be developed and improved through the use of an SE language. Section 3 examines the difference between systems modeling and model-based systems engineering (MBSE) and the implications of these differences. Section 4 presents agile systems engineering (ASE), supported by MBSE, as a practical approach to bridge the gap between the burdensome traditional SE processes and agile software development methods. The conclusions of the paper are in Section 5.

## The Language of Systems Engineering

The linguistic relativity hypothesis posits that the language an individual or culture speaks shapes the way they think and perceive the world (Wolff and Holmes 2011). While there is still debate in the linguistic world about the strength and scope of this hypothesis, there is a great deal of supporting evidence and many examples of its application. An example that is particularly relevant to this discussion has to do with the Pormpuraaw, an aboriginal group in Australia. In their language, Kuuk Thaayorre, there are no words for relative positions equivalent to "left" or "right" in English. Instead, all references to the positions of objects are based on the cardinal directions (north, east, south, and west). As a result, even young children of the Pormpuraaw exhibit exceptional ability to know what direction they are facing at all times, to a degree that some experts did not think was possible for humans. This phenomenon is not unique to sense of direction as similar language-based effects have been observed with color perception and math skills, among others. Cognitive psychologist Lera Boroditsky has shown that this phenomenon is strong evidence that language not only affects how an individual thinks and perceives the world but that individuals can develop cognitive ability based on the language they use (Boroditsky 2011).

Much like the difference between the natural language of individuals or cultures, different fields of study, professions, and other social groups have developed their own "languages" in the form of ontologies which are "a collection of standardized, defined terms and relationships between the terms" (Vaneman 2018). Using a domain-specific ontology allows group members to communicate with each other in a more direct and precise manner using a more meaningful vocabulary. Based on Boroditsky's research on the connection between language and cognitive ability, there is reason to believe that the adoption and understanding of a domain-specific ontology may improve the cognitive ability of an individual within that domain.

There have been efforts within the systems engineering community to codify a systems engineering ontology (Vaneman 2018; Van Ruijven 2013; Honour and Valerdi 2006). The purpose of this paper is not to argue for or against a single systems engineering ontology or what should or should not be contained in such an ontology. The reality is that, even in the absence of an authoritative ontology, systems engineering does have its own "language" based on decades of experience and research. Individuals who "speak" the language of systems engineering may develop cognitive abilities specific to doing the work of a systems engineer and shape the way they perceive and approach problems.

Abandoning the language of systems engineering introduces the risk of sacrificing its associated cognitive ability. Notably, the cognitive ability associated with language seems to be tied to the language itself, not specific practices or processes. Innovations and improvements to the application of SE can, and should, continue while preserving its language. The language of systems engineering will continue to evolve, as all languages do, but as it does, it should be built on the existing body of knowledge and experience. This is not to say that a system will be doomed to failure without this language, but the advantage of proven principles of systems engineering will be lost without it. Instead of abandoning SE, we need to strengthen the way that SE and its accompanying language is documented, communicated, and taught to harness the power of the linguistic relativism hypothesis to develop the cognitive ability to be better systems engineers.

## Demand for Effective MBSE Implementation

MBSE has emerged as an approach to systems engineering that specifically addresses the challenges associated with large, complex systems and managing rapid change inherent to compressed delivery timelines and agile software development practices. Despite the tendency to always refer to this approach by its acronym, it is important to recognize that MBSE is still fundamentally systems engineering. Though specific approaches vary, all of the processes and actions that have been done in SE traditionally should still be used in an MBSE approach. The primary difference from the traditional approach is that in an MBSE approach these processes are model-driven and the outputs are captured in a system model that can function as a single authoritative source of truth (ASoT) for SE data. As shown in Fig. 2, the exponentially increasing interactions of the traditional, document-centric approach can be reduced as all stakeholders are able to deliver and access SE data in the model directly.

This is how MBSE is typically presented on paper, but in practice there are challenges associated with MBSE adoption that have prevented its adoption to the degree that would be expected based on

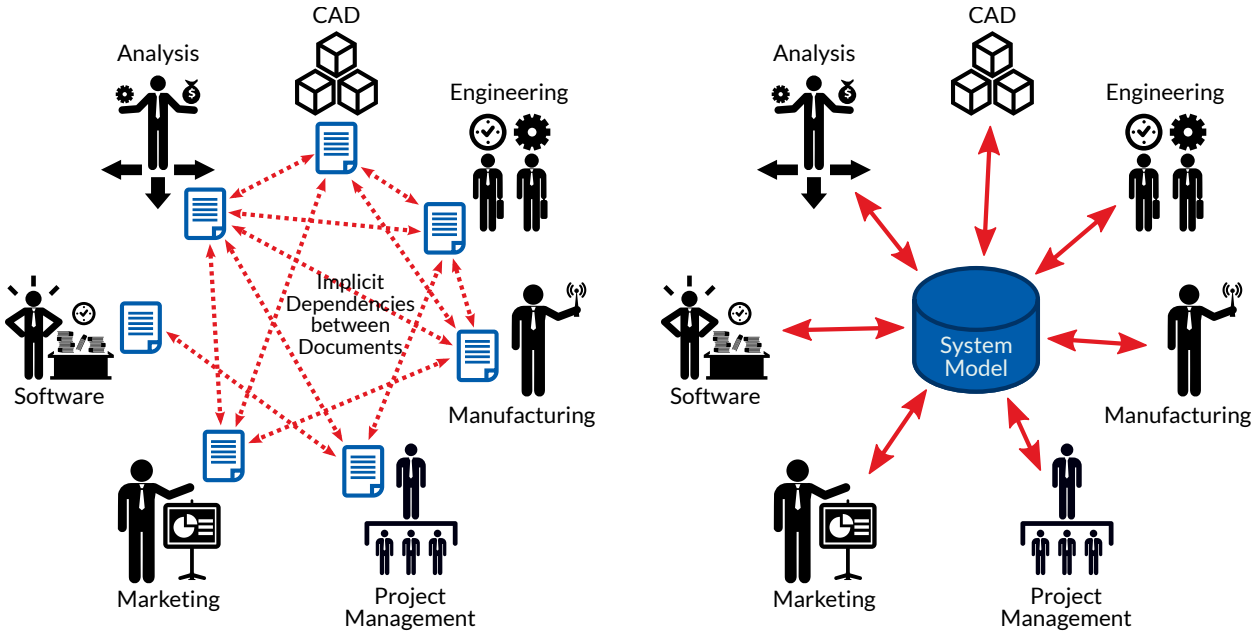


Figure 2. Illustration of traditional document-centric systems engineering and model-based systems engineering.

its documented benefits (Cameron and Adsit 2020; Madni and Sievers 2018; Chami and Bruel 2018). The diffusion of innovation theory describes the variability of the adoption rate of innovations generally (Rogers 2003), and may aid in understanding the adoption rate of MBSE specifically (Call and Herber 2022).

Another factor that may be affecting MBSE adoption and success is that there is still no widespread understanding of what exactly constitutes MBSE throughout the SE community, leading to ineffective MBSE implementations. A frequently cited definition of MBSE, provided by INCOSE, is "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" (Walden et al. 2015). Without an understanding of exactly what a "formalized application of modeling" entails or what it means to "support" systems engineering activities throughout the system life cycle, the scope as to what can be considered MBSE is not immediately clear.

The result can be illustrated using the same hypothetical organization referenced in the introduction. Because of the atrophied SE activity, as already described, integration challenges arise, and system quality suffers. MBSE is a popular topic and receives a lot of attention, so the chief engineer and program manager decide that they will "do MBSE". Expensive tool licenses are purchased, several engineers are trained on a modeling language and tool, and these newly trained engineers are assigned to a development team with the role of "modelers". Development work continues much as it has before, with the system modelers capturing architecture and design decisions as they are made

or implemented in a system model. There are occasional times that a potential problem is identified through the use of the model, but the model is used primarily as a way to document decisions that have already been made, not as a design tool. Over time modeling efforts wane as no one is really using the model, and it slows work down more than it helps. The model never delivers value to the organization, and everyone involved eventually accepts the fact that MBSE is a fad and begins looking for another path forward to solve their integration and quality challenges.

This hypothetical organization may be somewhat caricatured, but it raises the question of what constitutes an MBSE approach that has a likely chance of providing value to an organization? Delligatti builds on the INCOSE definition and identifies three pillars associated with MBSE — a modeling language, a modeling tool, and a modeling methodology (Delligatti 2013). The example organization clearly had a modeling language and tool, but no methodology was specified. A modeling methodology establishes the purpose of the modeling effort and identifies specific tasks and standards that will be used in the creation of the model. The importance of a modeling methodology suggests that the mere existence of a system model is not what provides value to an organization and does not constitute a true MBSE approach (Haskins 2011). Rather, it is the systems engineering processes that are used and the act of developing the model where the value is generated.

Vaneman supports the argument against the INCOSE definition of MBSE, making the point that it does not go far enough. His expanded definition addresses the ambiguity of the term "support" in the INCOSE definition. He defines MBSE as "the formalized application of modeling (static and dynamic) to support system design and analysis, throughout all phases of the system lifecycle, through the collection of modeling languages, structures, model-based processes, and presentation frameworks used to support the discipline of systems engineering in a model-based or model-driven context" (Vaneman 2018). This extension of the INCOSE definition specifies what is included in an MBSE approach. To describe how useful models are produced, this definition stipulates the use of a modeling language (as opposed to plain English), defined structures, model-based processes (that is, processes that are centered on the modeling efforts with results that are captured in the model), and the frameworks by which the modeling artifacts will be presented.

Some other additions that are worth highlighting is that this definition includes both static and dynamic models. Even if we consider traditional SE artifacts to be models, since they are abstract representations of the system of interest, they are still only static models. The dynamic models that are enabled by MBSE yield significant value to the SE process. Finally, the inclusion of systems engineering activities being model-driven adds further emphasis to the central role of models in an MBSE approach. Models are not created to capture decisions made using arbitrary processes. Instead, the models drive and inform new, modified systems engineering processes that are employed in an MBSE approach. This definition of MBSE is a valuable contribution to the understanding of what exactly MBSE entails, and its application could lead to improved MBSE outcomes.

## Agile Systems Engineering

The introductory section of this paper introduced a hypothetical organization that abandoned SE rigor in an effort to realize the benefits offered by an approach utilizing agile principles and methods.

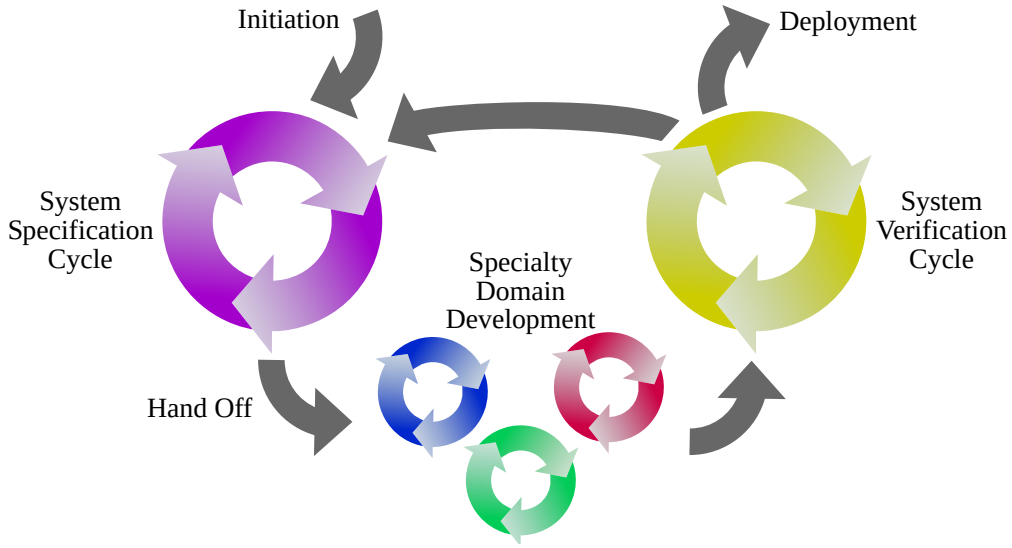


Figure 3. An agile systems engineering approach (adapted from Douglass 2015).

However, this choice between systems engineering rigor and agile methodology is a false dichotomy. There is no reason why a robust systems engineering approach cannot include agile principles. While most established agile methods apply specifically to software development, agile principles are generally related to being adaptive to change, iterative and incremental development, and being people-oriented (Abbas, Gravell, and Wills 2008).

The desire and need to apply agile methods to SE processes has given rise to the concept of agile systems engineering (ASE) (Walden et al. 2015). When referring to ASE, it is important to make a distinction between *agile systems* engineering and agile *systems engineering*. The former refers to the engineering of systems that could be described as agile in their ability to adapt and respond to change, while the latter refers to a systems engineering approach that incorporates agile principles that is used to engineer systems that themselves may or may not be agile (Haberfellner and Weck 2005). While the development of agile systems is vital in many instances, the focus of this paper is on SE approaches, and all future references in the paper to agile systems engineering are in reference to an agile *systems engineering* approach. A generic ASE approach that uses incremental and iterative development to respond to change is depicted in Fig. 3. Note that this figure of an iterative (and potentially incremental) SE approach explicitly depicts SE activities as cycles that provide inputs to and receives feedback from preceding and subsequent events, as opposed to discrete events seen in the traditional systems engineering "vee" model (Walden et al. 2015; Douglass 2015).

ASE does not refer to a specific approach as there is no single, commonly accepted definition of or process that can be authoritatively referred to as agile systems engineering. In fact, Kemp et al. have identified four distinct ways that the term agile SE is used (Kemp, Akroyd-Wallis, et al. 2016):

1. Agile System Development — Closest to an agile approach as understood in a software development context as it is focused on using operational feedback to carry out SE processes in response to rapid environmental and requirement changes.

2. Higher Tempo Conventional SE — Tailoring of conventional SE processes and tools to reduce the overall amount of time spent on SE in support of compressed system delivery timelines where there is a desire to employ agile methods, but the SE process must be clearly defined (e.g., safety-critical and/or highly regulated systems).
3. Agile SE Document/Model Development — An approach where users are engaged early on and agile principles are applied primarily to the development of the system specification (document or model) for hand off to other engineering disciplines.
4. Snake Oil — This is not really an agile approach at all, but an instance where agile principles are presented as an excuse to avoid systems engineering rigor or "adopted" to comply with an expectation or mandate with no understanding of how to do so or evidence of likely success. This is the approach utilized by the hypothetical organization from the introduction and is, unfortunately, applied far too often.

The concept of ASE is a natural and necessary progression for SE in a world where software is a dominant driver of modern systems. Because software is responsible for the implementation of high-level specifications that are the output of systems engineering processes and the relative ease of producing software compared to physical components, software size and complexity have necessarily outpaced the number of components and/or subsystems that contribute to systems engineering complexity. Because of this, the software development community has been addressing the challenges of managing this size and complexity for decades. However, as systems have continued to become more complex and software-centric, the systems engineering community is now facing many of the same problems associated with the level of complexity that the software development community encountered that led to the conception and formalization of agile software development key statements, principles, and processes. Systems engineers have the benefit of leveraging and adapting research on and development of agile methods that software developers have developed over the last 25+ years.

The development and maintenance of document-based SE artifacts present challenges in an ASE approach due to the pace and scope of required changes to those artifacts. However, MBSE and ASE are very complementary, and not only can but *should* coexist (Douglass 2015; Huss, Herber, and Borky 2023). Though they share many attributes, ASE and MBSE are not synonymous and are distinct specializations of a more generalized SE approach. A systems engineering approach for an organization could include both ASE and MBSE elements, ASE or MBSE elements, or contain neither ASE nor MBSE elements as illustrated in Fig. 4.

ASE is not new, and an ASE approach to developing a systems specification has been thoroughly elaborated by Douglass (Douglass 2015), among others (Haberfellner and Weck 2005; Dove and LaBarge 2014; Schindel and Dove 2016; Bonnet et al. 2015; Huss, Herber, and Borky 2023). Though the agile manifesto (Beck et al. 2001) was written for software development, Douglass has mapped its key statements and principles to systems engineering. While his approach is focused on the development of a system specification, he identifies some agile best practices that are particularly suited for an ASE approach utilizing MBSE (Douglass 2015):

- Incremental development of work products
- Continued verification of work products
- Executable requirements models

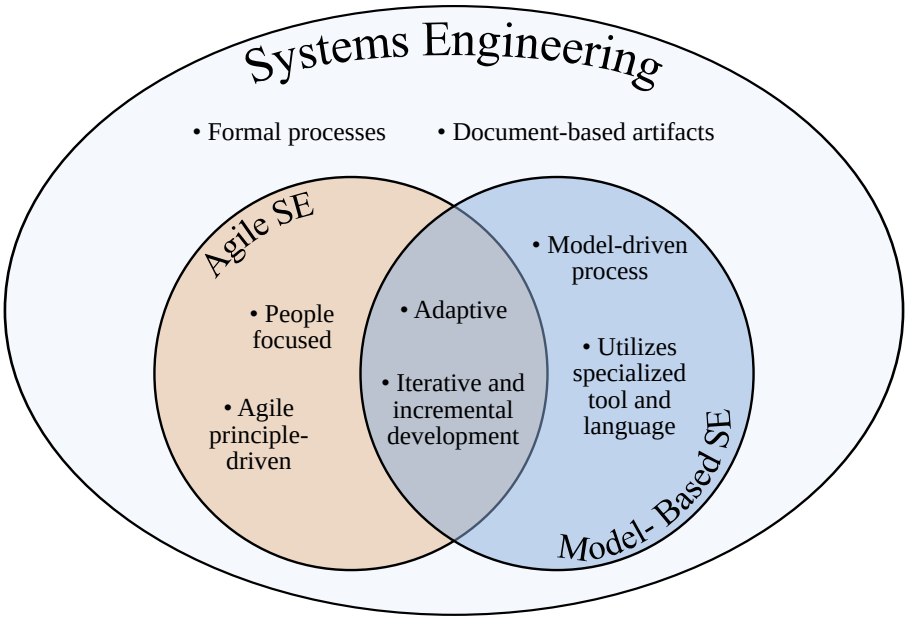


Figure 4. Relationships between systems engineering generally, agile SE, and MBSE.

- Model-based specification linked to a textual specification
- Model-based hand off to downstream engineering
- Dynamic planning

**Incremental development of work products** — When SE activities are strictly carried out serially, that is, subsequent activities are only begun once the preceding activity is completed, then critical decisions may be made on incomplete information. The alternative, agile approach would be to begin subsequent SE activities as soon as "just enough" of the preceding activity is completed. By working incrementally, lessons learned from subsequent activities can be fed back to preceding activities and changes made before additional, incorrect work is completed. An MBSE approach, with its strength in managing change, is well suited for this situation.

**Continued verification of work products** — The primary means of verifying work products in a document-based SE approach is through the use of reviews. This is not because there is no desire for a more robust verification process but primarily due to the absence of any other option. When SE work products are model-based, validation rules can be built into the modeling tool to ensure product quality, compliance with standards, and internal consistency. Component and subsystem behaviors specified in the model can be simulated to verify they yield the desired system behavior. This MBSE capability allows for identifying and resolving potential defects during the design phase before they are introduced into the realized system necessitating expensive rework to correct.

**Model-based specification linked to textual specification** — There are benefits (and sometimes requirements) to being able to demonstrate traceability from textual requirements to the system behaviors that satisfy those requirements and the structural elements to which those behaviors are allocated. With this traceability, stakeholders can be assured that all requirements are accounted for in the design and allow for system design trade space exploration before development costs are

incurred. An MBSE approach affords this traceability in a way that is prohibitively difficult in a document-based approach. Using any number of commercially available modeling tools and a robust modeling methodology, the process of creating, maintaining, and reporting on these traceability links can be somewhat natural and eventually trivial.

**Model-based hand off to downstream engineering** — Text-based requirements and specifications written in a natural language are often ambiguous, which can lead to misunderstanding when they are to be implemented. The use of a modeling language, like SysML, that has at least semi-formal semantics can help reduce some of this ambiguity. If downstream engineers are versed in the modeling language and organizational modeling standards and conventions, then communication can be more precise and the verified designs generated by the systems engineering team can be implemented more effectively. Again, when changes are required based on development work by downstream engineering, those changes are propagated throughout the specification more effectively and consistently using a model-based approach. This model-based hand off yields these same benefits upstream the right side of the vee to systems integration and verification and validation teams.

**Dynamic planning** — The ability to plan dynamically is one of the most powerful agile principles that can be applied to SE. Dynamic planning is in contrast to static, or as Douglass refers to it "ballistic planning", where a plan is made from the start of a project and then followed (or at least attempted to be followed) through project completion (Douglass 2015). With a dynamic planning paradigm, there is an acknowledgment by all the project stakeholders of the many unknowns at the beginning of a project. Any plans based on the information known at the project's outset are almost certainly flawed. General road maps and milestones are appropriate, but detailed planning should be conducted as incrementally as possible as more is learned about the system and its context through the SE processes. MBSE is a key enabler of dynamic planning when it comes to systems engineering work because this planning will result in iterations of systems engineering artifacts at a pace that cannot be effectively managed using a document-based approach.

It would appear that ASE, especially when it is paired with MBSE, may be able to effectively bridge the gap between the need to respond faster to changing environments and the benefits of robust SE. In spite of this, ASE has not received the same level of attention as MBSE. A cursory search through the proceedings of the INCOSE International Symposium from 2015 to 2022 shows an average of 4.6 papers published per year that included the term "agile systems engineering" in the abstract compared to 30 papers per year for "model-based systems engineering". These findings are summarized in Fig 5. Many MBSE publications address many of the same problems ASE can address, but an approach to these problems would likely benefit from more research into expanding the application of agile principles and ASE along with MBSE.

## Conclusions

Even though the practice of SE has enabled the development of many of the products and capabilities that characterize the modern world, there is pressure within some organizations to abandon traditional SE approaches. This is due, at least in part, to the role that software has played in drastically increasing the complexity of modern systems and the success agile methods have had in improving software development practices. Despite these pressures, it would be a mistake to disregard time-

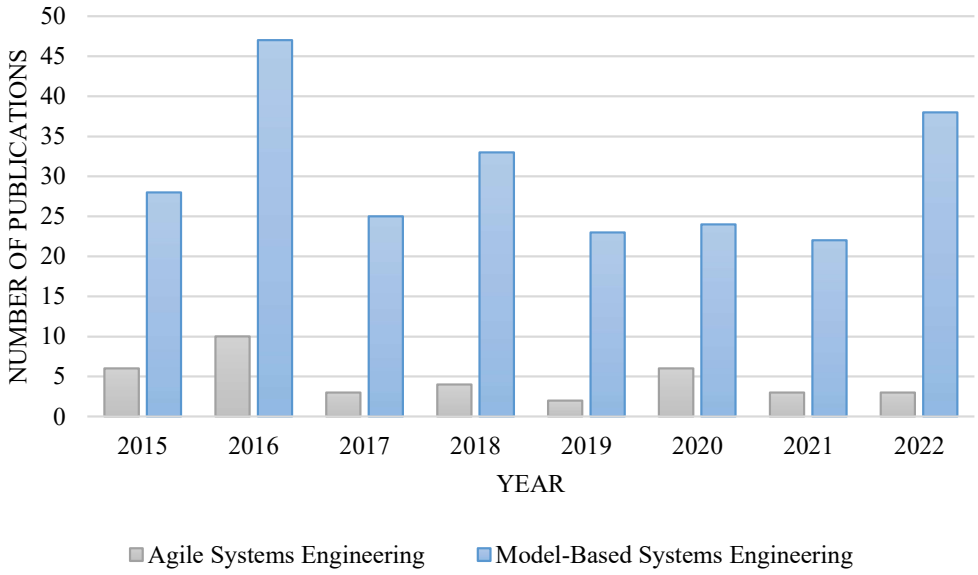


Figure 5. Comparison of publications about ASE compared to MBSE in INCOSE IS proceedings.

tested SE principles.

It is important to appreciate that the value of SE is not derived solely from the delivery of SE artifacts (document or model-based) like specifications, requirements, and architecture diagrams, even when those artifacts are properly tailored to a unique system context. Rather, value in SE is generated primarily from the processes that generate those artifacts. These processes are described by and utilize a unique SE language or ontology. The study of linguistics and the linguistic relativism theory has shown the power of language in developing and enhancing the cognitive abilities of its speakers in specific domains related to the language. The practice of SE can, and should, evolve to be more relevant to current challenges, but the "language'' of SE should be preserved to retain and enhance the cognitive abilities that can be gained through its use.

MBSE addresses many of the limitations of a document-based SE approach. Despite this, MBSE has not been adopted to the degree that would be expected based on its potential benefits and ongoing SE challenges. Again, when the focus of SE is on the resulting artifacts (in this case, a system model) instead of the processes and activities used to develop the system models, much of the value of SE is not realized. Too many MBSE implementation attempts are ineffective or fail because of a focus on systems modeling (the artifact) instead of model-based *systems engineering* (the process). A renewed emphasis on the importance of an MBSE methodology to focus and motivate modeling efforts can help to restore the SE to MBSE.

Additionally, an improved, more explicit definition of what constitutes an MBSE approach may improve the practice of MBSE and improve SE outcomes. Finally, the pressure within organizations to adopt agile methods in place of traditional SE should not be interpreted as an indictment of foundational SE principles. The tension between agile methods and traditional SE does expose the critical need to evolve and adapt the practice of SE when dealing with complex systems, especially

when those systems are software-centric. Agile systems engineering, or the application of agile principles to systems engineering, is a natural and appropriate evolution of SE in an agile world. The perceived conflict between agile methods and traditional SE is not due to SE principles and processes. Instead, it arises from the difficulty in managing and maintaining accurate and consistent document-based SE artifacts in the face of rapid change. An *effective* MBSE approach provides the processes and tools to manage SE artifacts in the face of rapid change. Thus, MBSE is a key enabler to ASE and enables the continued application of SE principles and processes with their associated benefits.

By harnessing the language of SE in improving and advancing the practice of MBSE-enabled ASE, organizations will be able to reap the benefits of agile methods *and* SE to deliver systems that meet their stakeholder needs in increasingly complex environments.

## References


Abbas, N., A. M. Gravell, and G. B. Wills (2008). "Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From?" In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by P. Abrahamsson et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 94–103. doi. 10.1007/978-3-540-68255-4_10.

Ammer, C. (1997). *The American Heritage Dictionary of Idioms*. Houghton Mifflin, p. 729.

Beck, K. et al. (2001). *Manifesto for Agile Software Development*. Accessed: 2022-12-14. URL: http://www.agilemanifesto.org.

Bonnet, S. et al. (2015). "Implementing the MBSE cultural change: organization, coaching and lessons learned." In: *INCOSE International Symposium* 25.1, pp. 508–523. doi. 10.1002/j.2334-5837.2015.00078.x.

Boroditsky, L. (2011). "How Language Shapes Thought." In: *Scientific American* 304.2, pp. 62–65. doi. 10.1038/scientificamerican0211-62.

Call, D. R. and D. R. Herber (Nov. 2022). "Applicability of the diffusion of innovation theory to accelerate model-based systems engineering adoption." In: *Systems Engineering* 25.6, pp. 574–583. doi. 10.1002/sys.21638.

Cameron, B. and D. M. Adsit (2020). "Model-based systems engineering uptake in engineering practice." In: *IEEE Transactions on Engineering Management* 67.1, pp. 152–162. doi. 10.1109/tem.2018.2863041.

Carson, R. S. (2013). "Can Systems Engineering be Agile? Development Lifecycles for Systems, Hardware, and Software." In: *23rd Annual International Symposium of the International Council on Systems Engineering, INCOSE 2013* 1.1, pp. 368–380. doi. 10.1002/j.2334-5837.2013.tb03001.x.

Chami, M. and J.-M. Bruel (2018). "A Survey on MBSE Adoption Challenges." In: *INCOSE International Symposium* 28.1, pp. 1463–1477. doi. 10.1002/j.2334-5837.2018.00561.x.

Delligatti, L. (2013). *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Pearson Education.

Douglass, B. P. (2015). *Agile Systems Engineering*. Elsevier. doi. 10.1016/c2014-0-02102-8.

Dove, R. and R. LaBarge (2014). "Fundamentals of Agile Systems Engineering – Part 2." In: *INCOSE International Symposium* 24.1, pp. 876–892. doi. 10.1002/j.2334-5837.2014.tb03187.x.

Dvorak, D. (2009). "NASA study on flight software complexity." In: *AIAA Infotech@Aerospace Conference*, p. 1882. doi. 10.2514/6.2009-1882.

Emes, M., A. Smith, and D. Cowper (2005). "Confronting an identity crisis - How to "brand" systems engineering." In: *Systems Engineering* 8.2, pp. 164–186. doi. 10.1002/sys.20028.

Haberfellner, R. and O. de Weck (2005). "Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering." In: *INCOSE International Symposium* 15.1, pp. 1449–1465. doi. 10.1002/j.2334-5837.2005.tb00762.x.

Hagen, C. and J. Sorenson (2013). "Delivering military software affordably." In: *Interpreting the Middle East*. Vol. 42. 2. Routledge, pp. 30–34. doi. 10.4324/9780429499708-7.

Haskins, C. (2011). "A historical perspective of MBSE with a view to the future." In: *INCOSE International Symposium* 21.1, pp. 493–509. doi. 10.1002/j.2334-5837.2011.tb01220.x.

Hoban, F. T. and W. M. Lawbaugh (1993). *Readings in Systems Engineering*. Vol. 6102. NASA Scientific and Technical Information Program.

Honour, E. C. (2004). “Understanding the Value of Systems Engineering.” In: *INCOSE International Symposium* 14.1, pp. 1207–1222. doi. 10.1002/j.2334-5837.2004.tb00567.x.

Honour, E. C. and R. Valerdi (2006). “Advancing an ontology for systems engineering to allow consistent measurement.” In: *Systems Engineering* 11.3, pp. 221–234. doi. 10.1002/sys.20096.

Huss, M., D. R. Herber, and J. M. Borky (Apr. 2023). “An Agile Model-Based Software Engineering Approach Illustrated through the Development of a Health Technology System.” In: *Software* 2.2, pp. 234–257. doi. 10.3390/software2020011.

Kemp, D., K. Akroyd-Wallis, et al. (2016). “ “Fifty Shades of Agile”: AN ANALYSIS OF DIFFERENT PERSPECTIVES OF AGILE SYSTEMS ENGINEERING.” In: *INCOSE International Symposium* 26.1, pp. 691–712. doi. 10.1002/j.2334-5837.2016.00187.x.

Kemp, D., R. Beasley, and S. Williams (2015). ““Suits you sir! – choosing the right style of SE before tailoring to fit”.” In: *INCOSE International Symposium* 25.1, pp. 1245–1262. doi. 10.1002/j.2334-5837.2015.00127.x.

Laanti, M. (2014). *Characteristics and Principles of Scaled Agile. XP 2014 International Workshops*. Ed. by T. Dingsøyr et al. Cham: Springer International Publishing, pp. 9–20.

Madni, A. M. and M. Sievers (2018). “Model-based systems engineering: motivation, current status, and research opportunities.” In: *Systems Engineering* 21.3, pp. 172–190. doi. 10.1002/sys.21438.

Potocki de Montalk, J. (1993). “Computer software in civil aircraft.” In: *Microprocessors and Microsystems* 17.1, pp. 17–23. doi. 10.1016/0141-9331(93)90089-p.

Rogers, E. M. (2003). *Diffusion of Innovations*. 5th. Free Press, p. 512.

Rousseau, D. (July 2018). “A Framework for Understanding Systems Principles and Methods.” In: *INCOSE International Symposium* 28.1, pp. 1170–1189. doi. 10.1002/j.2334-5837.2018.00541.x.

Schenker, A., T. Smith, and W. Nichols (2022). “Measuring Digital Engineering Effectiveness for Embedded Computing Systems. Measuring the Effectiveness of Aesthetically Relevant UI Design for New Technologies.” In: *Advances in Systems Analysis, Software Engineering, and High Performance Computing*. IGI Global, pp. 86–110. doi. 10.4018/978-1-7998-9121-5.ch005.

Schindel, B. and R. Dove (2016). “Introduction to the agile systems engineering life cycle MBSE pattern.” In: *INCOSE International Symposium* 26.1, pp. 725–742. doi. 10.1002/j.2334-5837.2016.00189.x.

Sillitto, H. et al. (July 2018). “What do we mean by “system”? - System Beliefs and Worldviews in the INCOSE Community.” In: *INCOSE International Symposium* 28.1, pp. 1190–1206. doi. 10.1002/j.2334-5837.2018.00542.x.

Van Ruijven, L. C. (2013). “Ontology for systems engineering.” In: *Procedia Computer Science* 16, pp. 383–392. doi. 10.1016/j.procs.2013.01.040.

Vaneman, W. K. (2018). “Evolving Model-Based Systems Engineering Ontologies and Structures.” In: *INCOSE International Symposium* 28.1, pp. 1027–1036. doi. 10.1002/j.2334-5837.2018.00531.x.

Walden, D. D. et al., eds. (2015). *Systems engineering handbook: A guide for system life cycle processes and activities. A Guide for System Life Cycle Processes and Activities*. 4th. Wiley.

Wolff, P. and K. J. Holmes (2011). “Linguistic Relativity.” In: *Wiley Interdisciplinary Reviews: Cognitive Science* 2.3 (3), pp. 253–265. doi. 10.1002/wcs.104.

# Biography

**Daniel R. Call** is a Ph.D. candidate in the Systems Engineering program at Colorado State University and a Systems Engineer for the United States Air Force. His work has included test and evaluation, requirements engineering, early life cycle systems engineering, and integration of MBSE with software development. His research interests are in MBSE, development and usage of reference models and architectures, and the application of social sciences to systems engineering. He earned his B.S. in Mechanical Engineering from Brigham Young University in 2009 and his M.S. in Systems Engineering from the Naval Postgraduate School in 2018.

**Daniel R. Herber** is an Assistant Professor in the Systems Engineering Department at Colorado State University. His research interests are in the areas of computational design, model-based systems engineering, digital engineering, design optimization, and combined physical and control system design (control co-design) concentrated around the development of novel theory and tools for integrated design methods conducive to emerging and dynamic engineering systems. His work has involved several engineering application domains taking a systems perspective on development and design, including the design of offshore wind/wave energy systems, thrust reversers, carbon capture systems combined with thermal storage and natural gas plants, thermal management networks for aircraft, and strain-actuated solar arrays for reorienting spacecraft. He studied at the University of Illinois at Urbana-Champaign, earning his B.S. in General Engineering in 2011 and his M.S. and Ph.D. in Systems and Entrepreneurial Engineering in 2014 and 2017, respectively. He held a postdoctoral position (2018-2019) with the NSF Engineering Research Center for Power Optimization for Electro-Thermal Systems (POETS).